A Hardware Accelerated Robot Middleware Package for Intelligent Processing on Robots

A Hardware Accelerated Robot Middleware Package for Intelligent Processing on Robots

Yutaro Ishida*, Takashi Morie* and Hakaru Tamukoh*
*Graduate School of Life Science and Systems Engineering
Kyushu Institute of Technology, Kitakyushu, Fukuoka 808-0196
Email: ishida.yutaro954@mail.kyutech.jp

***Abstract*—Service robots require implementation of intelligent processing, e.g., image processing. However, the computational resources of standard PCs typically used in service robots are not sufficient for such processes. Furthermore, robot middleware is widely used in many robots because such systems facilitate integration and are suitable for rapid prototyping. We propose a "connective object for middleware to accelerator (COMTA)," which is a processing system that uses hardware accelerators, i.e., field programmable gate arrays (FPGAs), and robot middleware. Users can access the FPGAs in the proposed system via middleware interfaces; thus, complex internal circuits are not required. For human tracking using image processing, the proposed system can automatically generate from a single configuration file. The proposed system performs 3.3 times more efficiently relative to computation than standard PCs in robots.**

I. Introduction

In recent years, service robots have become increasingly common. "Exi@" [1], a home service robot we developed for RoboCup@Home [2], [3] competition, is shown in Fig. 1. RoboCup@Home, an annual competition first held in 2006, is the largest worldwide competition for home service robots. In this competition, robots perform practical applications in residential and public environments, e.g., acting as a waiter in a restaurant. To realize such applications, robots are equipped with sensors and actuators that approximate the human senses and motor functions (Fig. 1).

A block diagram of the software in our home service robot is shown in Fig. 2. The software consists of perception and control units to process sensory data and control actuators, respectively. In these units, a significant number of intelligent processes run simultaneously. However, in ordinary robot systems, such software run on a laptop PC connected to the robot. Thus, computing resources are always insufficient. In addition, recognition technologies that use deep learning have been implemented on most robots [4], [5]. However, such technologies incur significant calculation costs, and real-time processing on a laptop PC is difficult.

Furthermore, developing intelligent processing is becoming complex, as many different types of software and hardware must be integrated. For example, robotic engineers must deal with relatively simple device drivers and much more complex perception-level software. This requires multiple technologies and is extremely time-consuming. Therefore, robot middleware that supports open sources, large-scale software integration and prototyping is widely used in robots [6]–[11].

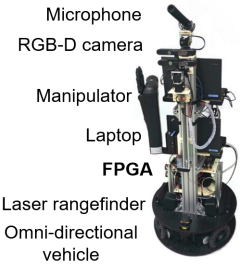


Fig. 1. Home service robot "Exi@" hardware

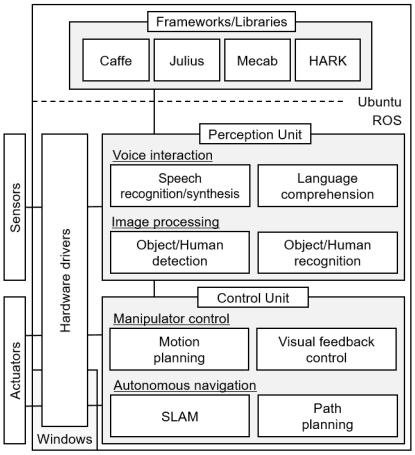


Fig. 2. Home service robot "Exi@" software

For these reasons, robots require an efficient (i.e., high-speed computation and low-power consumption) processing system that exploits accelerators with dedicated intelligent processing architecture. Expected applications of the system are robots ' perception such as image processing and object recognition by deep learning. In addition, the system must satisfy robot hardware requirements and should be easy to integrate with robot middleware.

Thus, we propose an efficient processing system that uses field programmable gate arrays (FPGAs) that can be integrated with the robot middleware. The proposed system is described in the following sections, and experimental results show the effectiveness of a practical application using the proposed system.

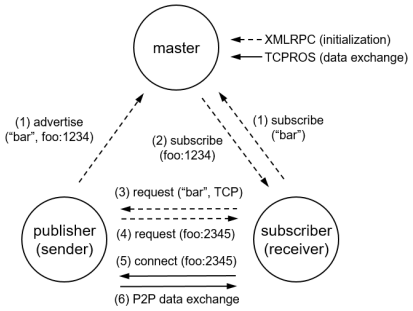


Fig. 3. Typical ROS Interface

II. Related Work

A. Robot middleware

Middleware mediate between an operating system and an application, and provides data communication/management and debugging functions. In recent years, several robot middleware has been developed, e.g., robot operating system (ROS) [6][7], OpenRTM-aist [8][9] and V-Sido OS [10].

ROS has many users worldwide [11]; many research institutes and companies have adopted it. In addition, many robotic systems ranging from research to commercial robots, are implemented using ROS. One of the reasons why ROS has attracted attention is its ease of integration using its own unified interfaces.

Figure 3 shows typical ROS interfaces related to a publish/subscribe messaging model. Note that ROS interfaces are based on a network connection. ROS has a master that manages the entire system. For the master, the publisher and subscriber register the request for data exchange using namespace (Fig. 3(1)). If the name matches, data exchange begins with a peer-to-peer (P2P) connection (Fig. 3(2-6)). Therefore, users only need to manage the namespace for automatic system integration.

Thus, it is important to automatically activate a processing system with an accelerator via the ROS interfaces for robot engineers to use the accelerator easily.

B. Accelerators suitable for robots

Graphics processing units (GPUs), application specific integrated circuits (ASICs), and FPGAs are all considered to be accelerator devices. GPUs enable parallel computation by exploiting many computing cores, and an ASIC is an integrated circuit (IC) chip optimized for a specific application. In addition, FPGAs are small size, reconfigurable digital circuits that operate at high speeds and with low-power consumption.

According to benchmark results obtained using the convolutional neural network [13], FPGAs are superior to embedded CPUs and mobile GPUs relative to both calculation speed and calculation performance per unit power. In addition, FPGAs generate less heat due to their low power consumption; thus, they are reliable under thermal considerations in embedded environments. However, the development man-hours required to implement them are significantly greater than CPUs and GPUs [14].

ASICs have an advantage compared to FPGAs in terms of computing performance and power consumption. In Addition, for mass-produced applications, ASICs are also advantageous relative to cost compared to FPGAs. However, FPGAs can be reconfigured; thus, they can be used to update an application even after it has been implemented in a robot.

In addition, robots generally use a limited power source, such as a battery; thus, it is difficult to implement GPUs, which have high power requirements, in robots. Furthermore, intelligent processing are evolving; thus, functions that cannot be updated after implementation in a robot become obsolete.

For these factors, FPGAs are considered to be suitable accelerators for robots. However, the number of development man-hours remains a problem. Therefore, it is important to propose a high-affinity robot middleware interface that connects with FPGAs, which would reduce the workload burden for both the circuit and robot engineers.

C. Accelerator applications

A previous study has applied FPGAs in hardware/software (hw/sw) complex system [15] that combines hw (an FPGA) and sw (a CPU). High-speed parallel computation is performed for processes that FPGAs can execute efficiently such as signal processing. On the other hand, processes that FPGAs cannot execute efficiently, such as complex conditional branches, are performed by CPUs. As a result, this combines the advantages of FPGAs and CPUs, which enables high-speed operation and reduces the power consumption of the entire system. Recently, the system on chip (SoC), which is an IC that integrates an embedded CPU and an FPGA on a single chip, has been released [17], and has shown a high degree of attention.

Another study has integrated a hw/sw complex system and robot middleware, i.e., ROS compliant FPGA components [12]. In that study, ROS was implemented in the CPU on the SoC and intelligent processing was implemented in the FPGA on the SoC. However, from a computing load perspective, it is unrealistic to compute intelligent processing installed in service robots using only the SoC. Therefore, it is important to propose a system that can be integrated with ROS installed on PCs.

In addition, integration of ROS and FPGAs for real-time control has been reported [16]. This research focused on latency for control problems. However, to accelerate intelligent processing on service robots, we should consider not only latency but also throughput.

III. Hardware accelerated ROS package

We propose a connective object for middleware to accelerator (COMTA), as shown in Fig. 4, as a hardware accelerated intelligent processing system for robots.

A. ROS specialized hardware accelerator model

COMTA automatically constructs a processing system that including a hardware accelerator from ROS space. A laptop PC (Fig. 4 left) and a hw/sw complex system (Fig. 4 right) are used in this processing system. The hw/sw complex

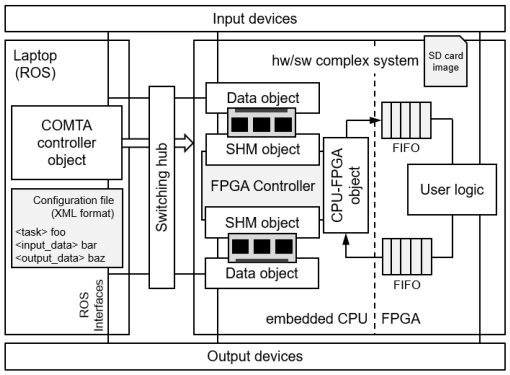


Fig. 4. Block diagram of proposed system（COMTA）

system is activated using an image file on an SD card. The sensors and actuators employ existing device drivers installed on an embedded CPU in the hw/sw complex system. Here, intelligent processing is performed by a dedicated architecture (referred to as user logic) implemented on an FPGA.

We propose to automatically generate the hardware accelerated intelligent processing system and its specialized hardware accelerator via the ROS interfaces, which is possible by using a block of programs called an "object".

A COMTA controller object is the core of the system and activates the hw/sw complex system. When it is defined and given by a tiny configuration file (Fig. 4) via the ROS interface, the objects in the hw/sw complex system are activated according to the following flow: (1) a data object is activated for data exchange with external devices or ROS on the laptop, and (2) a SHM and CPU-FPGA objects are activated for internal data exchange in the hw/sw complex system.

The user logic is connected via a FIFO interface; therefore, circuits can be optimized using HDL, a circuits can be generated by high-level synthesis, and any circuits can be connected. In addition, by implementing the FIFO interface, ROS users can use these circuits without considering their internal structures.

Consequently, the hardware accelerator can be handled simply like the previous ROS system. For example, for typical ROS navigation, users are simply required to introduce hardware (Laser rangefinder) into their system and download the software [18]–[20] required to run a navigation system. Similarly, users of the proposed hardware accelerated ROS package only need to implement an FPGA board and download the hardware accelerated ROS package from the Internet.

B. Shared memory communication model

The computational resources of the embedded CPU are very limited; thus, data exchange on the embedded CPU requires a communication model with a smaller computational load than that of the ROS interface shown in Fig. 3. Therefore, rather than using the ROS interface, we propose a shared memory communication model. This model exchanges data using only internal memory; thus, the computation load is less than that of the ROS interface, which exchanges data via a network.

IV. Experiment

To demonstrate the effectiveness of the proposed system, we implemented a human tracking application on a home service robot. The flow of the application is described as follows.

1. Acquire a depth image from an RGB-D camera using OpenNI [21].
2. Generate a binary image from the depth image with a threshold distance.
3. Extract region of interest images from the binary image using sliding windows.
4. Extract multi-resolution co-occurrence histograms of oriented gradients (MRCoHOG) features [22] from the region of interest images.
5. Determine whether the features are a human or not using a real AdaBoost [23].
6. Control the wheel base of the robot to approach the target human.
7. Set the distance of the human region as the threshold for generation of the next binary image.

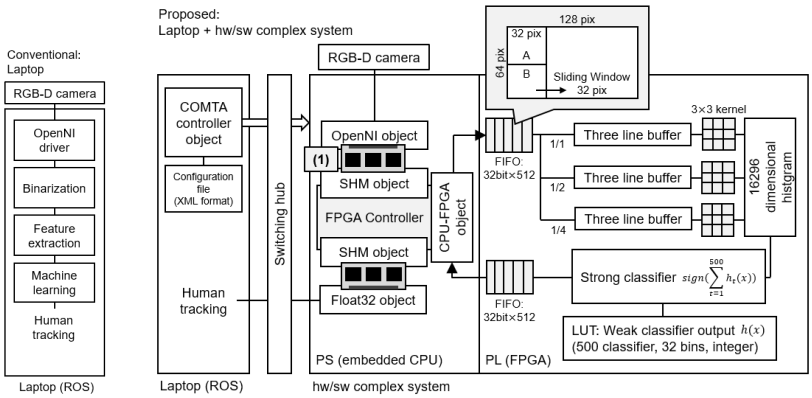
Figure 5 shows a block diagram of the system constructed for the experiment. Note that Fig. 5(a) shows the a conventional system. In this system, all operations (i.e., sensing to control operations) were performed on a laptop PC.

Figure 5(b) shows the proposed system. Here, operations were performed using a laptop and the hw/sw complex system. First, based on the behavior of the hw/sw complex system which is defined in the XML format, the COMTA controller object activates the embedded CPU's software. Then, the operation flow 1 is executed using an OpenNI object. At this time, the distance image is sent to an FPGA controller via a shared memory. Operation flows 2 and 3 are executed, on the FPGA controller, and the region of interest images are written to FIFO via the CPU-FPGA object. In the FPGA, the region of interest images are sent to three line buffers at the original, half, and quarter resolutions. The buffered images are calculated luminance gradients using the upper, lower, left and right of the 3×3 kernel, and vote on a histogram block. The histogram is evaluated by the output of weak classifiers implemented in LUT. Finally, the result is sent to the embedded CPU via FIFO and sent to the laptop via shared memory. Thus, the laptop is only used to activate the hw/sw complex system and control of the wheel base of the robot, and the computational load of the intelligent processing is offloaded to the hw/sw complex system.

In this experiment, we used a Xilinx XC7Z020 SoC and its evaluation board [17], [24]. To synthesize the internal circuit of the FPGA, we used a Xilinx Vivado HLS 2016.2 which is a high level synthesis tool [25]. The synthesis results shows that 8% Flip Flops and 31% Look Up Tables of available resources were used; whereas, no DPS and Block RAM were used.

A. Evaluation of embedded CPU internal communication

Data exchange in the embedded CPU is shown in Fig. 5(b)(1). We compare the performance of the conventional ROS interfaces and the proposed shared memory model. In



(a) (b)

Fig. 5. Block diagram; (a) conventional and (b) proposed systems

TABLE I
EMBEDDED CPU INTERNAL COMMUNICATION RESULTS

Method	Send/ Receive	Average frame rate [fps]	
		QVGA	VGA
ROS interfaces	Send	309	65
	Receive	244	60
Shared memory	Send	1261703	822011
	Receive	70550	77393

TABLE II
EXECUTION RESULTS OF THE PERSON TRACKING APPLICATION

	Conventional Core i5 5200U 2.2GHz	Proposed ARM Dual Core 667MHz/FPGA
Frame rate [fps]	29.06	17.25
Power consumption [W]	26	4.7
Efficiency [fps/W]	1.12	3.69

the former, data exchange is performed via a local network inside the embedded CPU, and the latter method is often used for inter-process communication with the data exchange performed using memory. In this experiment, QVGA/VGA, RGB 8 bit, three-channel images were exchanged. In addition, we measured the average frame rate.

The results are shown in Table I. As can be seen, the shared memory can exchange data at a high speed with a smaller calculation load.

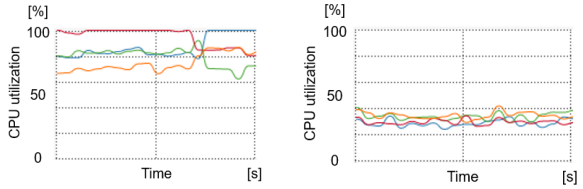
B. Evaluation of the proposed system using robot application

As mentioned previously, we compared the performances of the conventional and proposed systems using a human tracking application. In this experiment, frame rate, power consumption, power efficiency and CPU utilization were measured.

The experimental results are shown in Table II. In the conventional system, the processing frame rate was the same as that of the RGB-D camera. However, the conventional system consumed 26 W and its power efficiency is low.

In the proposed system, the processing frame rate was less than that of the conventional system. Here, operation flow 2 and 3 which are implemented in the embedded CPU, caused a reduction in operational speed. However, the proposed system consumed only 5 W, and the power efficiency was 3.3 times better than that of the conventional system.

The experimental results in terms of CPU utilization are shown in Fig. 6. As can be seen, the CPU utilization of the proposed system was less than that of the conventional system, which demonstrates that the heavy load incurred by person tracking was offloaded to the proposed system.



(a) Conventional (b) Proposed

Fig. 6. Results of CPU Utilization

V. CONCLUSION

We have proposed COMTA hardware accelerated robot middleware package. The proposed system can automatically generate an intelligent processing system using a tiny configuration file in ROS space. In addition, we focused on communication and proposed a shared memory communication model. The experimental results indicate that the communication of the proposed system has better throughput and latency than ROS interfaces. In a person tracking application, the power efficiency of the proposed system was 3.3 times better than that of a general PC system. In the future, we will improve the efficiency of intelligent processing, and implement further intelligent processing using deep neural networks.

ACKNOWLEDGMENT

This research is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and partially supported by JSPS KAKENHI grant number 17H01798.

REFERENCES

[1] S. Hori, Y. Ishida, Y. Kiyama, Y. Tanaka, Y. Kuroda, M. Hisano, Y. Imamura, T. Himaki, Y. Yoshimoto, Y. Aratani, K. Hashimoto, G. Iwamoto, H. Fujita, T. Morie and H. Tamukoh, "Hibikino-Musashi@Home 2017 Team Description Paper," The Computing Research Repository, arXiv:1711.05457 [cs.RO], 2017.

[2] T. Wisspeintner, T. van der Zant, L. Iocchi and S. Schiffer, "RoboCup Home: Scientific Competition and Benchmarking for Domestic Service Robots," Interaction Studies, Vol. 10, Issue 3, pp. 392-426, 2009.

[3] L. Iocchi, D. Holz, J. Ruiz-del-Solar, K. Sugiura, T. Zant, "Analysis and results of evolving competitions for domestic and service robots," Artificial Intelligence, Vol. 229, pp. 258–281, 2015.

[4] J. Redmon, S. K. Divvala, R. B. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788, 2016.

[5] Z. Cao, T. Simon, S. Wei and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[6] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "ROS: an open-source Robot Operating System," ICRA Workshop on Open Source Software, 2009.

[7] (2017, Nov.) ROS Wiki. [Online]. Available: http://wiki.ros.org/

[8] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku and W. Yoon, "RT-middleware: distributed component middleware for RT (robot technology)," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3555-3560, 2005.

[9] (2017, Nov.) OpenRTM-aist official website. [Online]. Available: http://www.openrtm.org/openrtm/ja/content/openrtm-aist-official-website

[10] (2017, Nov.) V-Sido OS. [Online]. Available: https://www.asratec.co.jp/v-sido-os/

[11] (2017, Nov.) ROS Community Metrics Report 2015. [Online]. Available: http://download.ros.org/downloads/metrics/metrics-report-2015-07.pdf

[12] K. Yamashina, T. Ohkawa, K. Ootsu and T. Yokota, "Proposal of ROS-compliant FPGA Component for Low-Power Robotic Systems -case study on image processing application-," International Symposium on Foundations and Practice of Security, The Computing Research Repository, abs/1508.07123, 2015.

[13] H. Yonekawa and H. Nakahara, "On-Chip Memory Based Binarized Convolutional Deep Neural Network Applying Batch Normalization Free Technique on an FPGA," IEEE International Parallel and Distributed Processing Symposium Workshops, pp. 98-105, 2017.

[14] K. Benkrid, A. Akoglu, C. Ling, Y. Song, Y. Liu and X. Tian, "High performance biological pairwise sequence alignment: FPGA versus GPU versus cell BE versus GPP," International Journal of Reconfigurable Computing, Vol. 2012, pp. 1-15, 2012.

[15] H. Tamukoh and M. Sekine, "Design of Networked hw/sw Complex System using Hardware Object Model and Its Application," Proc. of 39th Annual Conference of the IEEE Industrial Electronics Society, pp. 2250-2255, 2013.

[16] A. B. Lange, U. P. Schultz and A. S. Soerensen, "Towards Automatic Migration of ROS Components from Software to Hardware," The Computing Research Repository, arXiv:1407.7560 [cs.RO], 2014.

[17] (2017, Nov.) Zynq-7000 All Programmable SoC. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[18] (2017, Nov.) hokuyo_node. [Online]. Available: http://wiki.ros.org/hokuyo_node

[19] (2017, Nov.) amcl. [Online]. Available: http://wiki.ros.org/amcl

[20] (2017, Nov.) mode_base. [Online]. Available: http://wiki.ros.org/move_base

[21] (2017, Nov.) OpenNI 2 Downloads and Documentation. [Online]. Available: https://structure.io/openni

[22] S. Iwata and S. Enokida, "Object Detection Based on Multiresolution CoHOG," International Symposium on Visual Computing, pp. 427-437, 2014.

[23] R. E. Shapire and Y. Singer, "Improved Boosting Algorithms Using Confidencerated Predictions," Machine Learning, Vol. 37, pp. 297-336, 1999.

[24] (2017, Nov.) Xilinx ZedBoard : http://zedboard.org/product/zedboard.

[25] (2017, Nov.) Xilinx. [Online]. Available: https://www.xilinx.com/